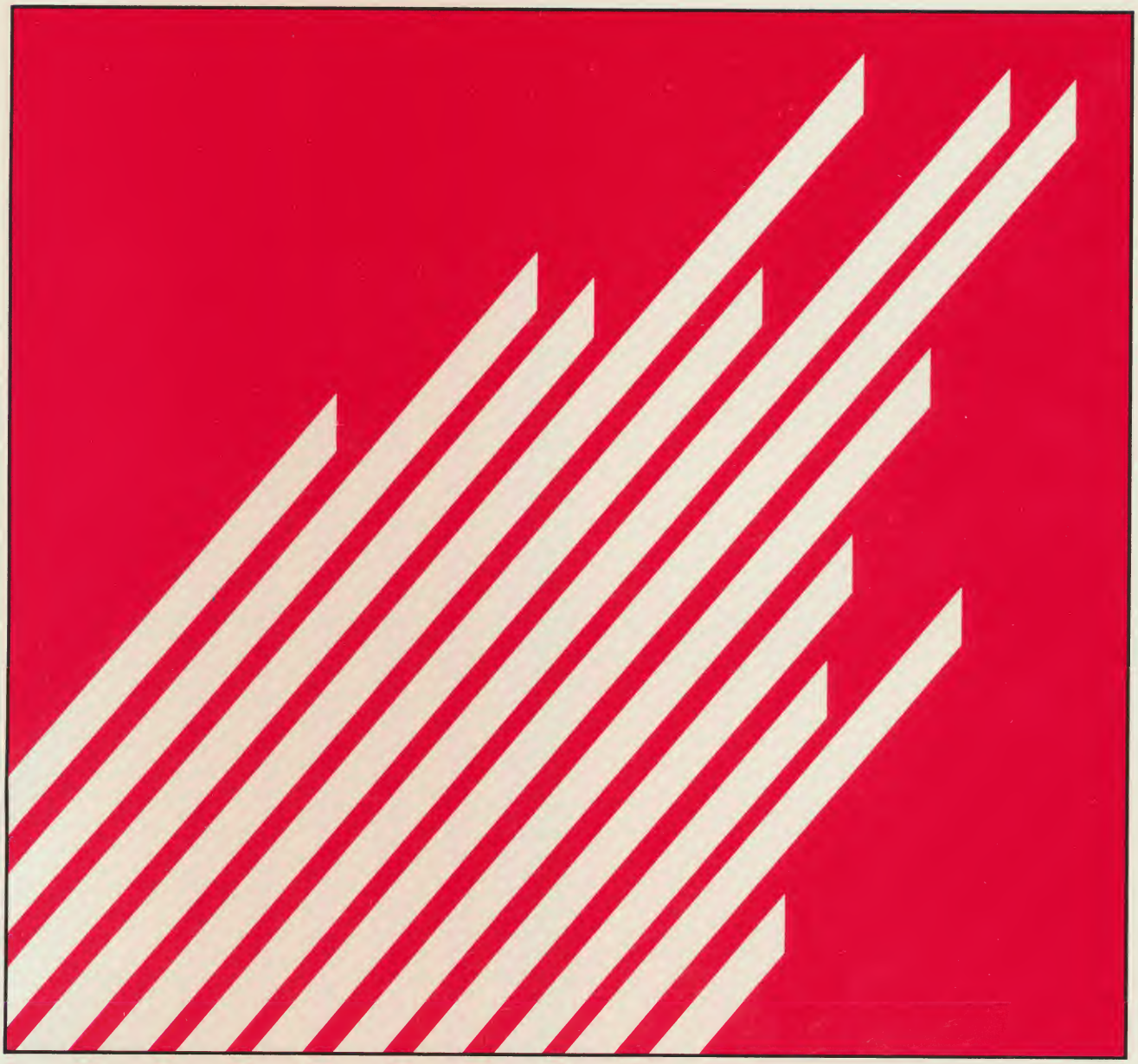
DL/I and SQL/DS Migration and Coexistence Guide



IBM®

GH21-1084-00

DL/I and SQL/DS Migration and Coexistence Guide

Document Number GH21-1084-00

pletely redesign and redevelop applications, relational technology may be considered for those systems. In this environment, there is the need to maintain DL/I and SQL/DS data, and applications using both types of data. The facilities discussed below in Chapter 10, "Coexistence Strategies" on page 161 should make it easier for you to develop and maintain these applications.

Use Application Packages Based On Relational Database: There are many excellent application packages today which support, or in some cases require, relational database management systems. Some of these systems (such as IBM's Communications Oriented Production Information and Control System - "COPICS") support either relational or hierarchic database systems. There are, of course, many advantages to acquiring an application package, such as avoidance of development and maintenance costs.

Some application packages are "transparent" with regard to the database management system upon which they execute. By "transparent," we mean that the application has the same look and feel, and provides the same information processing and delivery, regardless of whether the underlying data is stored in SQL/DS, DL/I, or other DBMS or data management system (such as VSAM). The advantage of a "transparent" application is that the ultimate users of the application (who are not usually information-systems professionals) need not learn new interfaces, screens, report formats, and procedures.

Other application packages are significantly different in the relational implementation, or may exist only on relational platforms. Often, the information delivery capabilities of these applications is significantly improved over older implementations. For example, some applications exploit not only relational database technology but graphical user interfaces and cooperative processing.

Migrate Selected Applications To Relational: Many organizations have selected one application or group of applications and migrated both the associated data and programs from non-relational to a relational database system. Though this may involve somewhat more effort than using relational for query of extracted data, or for support of an application package, the benefits may also be greater. The best type of application for this approach is most often one that is of medium complexity and criticality (neither trivial nor of a "bet your business" nature), and where the biggest need is for improved end-user self-sufficiency, access to the data, and/or additional reporting requirements.

Such a pilot project may be valuable for assessing the potential productivity and performance benefits of relational database management, with a limited and controllable investment of resources. As with the alternatives described above, there would be a requirement to maintain data and applications in both DL/I and SQL/DS for some period of time.

Migrate All Applications And Data To Relational: This alternative is not usually feasible if you have many applications and databases today using DL/I. It is usually more realistic to select only those applications for which maintainability could be improved with relational technology, and for which there is greatest demand for access to associated data via end user decision support tools based on relational technology.

Coexistence Considerations: There are many advantages to a strategy of coexistence of DL/I and SQL/DS over a fairly long term. The two products were designed from the beginning to coexist. Coordinated logging, locking, and

recovery; two-phase commit protocol; and operations procedures and commands provided with the products, all are intended to facilitate coexistence. There are some applications for which performance requirements will indicate that they should continue to use DL/I. Other applications will benefit from the flexibility of data design change, or end-user access capabilities, or application maintainability improvements associated with relational database management. Many applications today use data from both DL/I and SQL/DS. As DL/I and SQL/DS are complementary and well-integrated products, each application or group of applications may be evaluated on its own merits as to which database system(s) should be used.

2.2 Migration Techniques

If the decision is made to migrate one or more application systems to a relational database environment, then for each such application you must evaluate which technical methods are best suited for the applications and the associated data. Note that if you decide to implement one of the first three strategies described above - extract for query and reporting, develop new applications in relational without migrating existing applications, and/or purchase a package based on relational database management - then you may not need to be concerned with program migration. You would still need to consider the interaction of these new applications (and their data) with existing data. So in that case the data-modelling issues such as normalization would still need to be addressed.

If you choose to migrate programs and data, there are some tools available, either as products or in conjunction with service offerings, that may assist you. These are described in Appendix C, "Sources of Assistance" on page 181.

One-For-One Conversion: This method would initially appear to be the simplest. Its advantages are that it lends itself most easily to automation, and the applications function on the new system (e.g. SQL/DS) exactly as they did on the old (e.g. DL/I). Thus the users of these applications will not have to learn new procedures - in fact, they may not even be aware of the change of DBMS. The disadvantages are:

- Since each (or most) of the segment-types become an SQL/DS table, if there are repeating groups, undefined or self-defined portions of the data record, or redundant records, these are perpetuated in the relational design. This will not improve the database design for ease of end user access or maintainability.
- Since all DL/I calls in the programs are migrated (or more precisely, converted) one-for-one to SQL statements that usually operate on a single row, and procedural code dealing with navigation is not eliminated, the applications may not perform as well as if they had been redesigned to take advantage of the powerful capabilities of relational such as set processing and predicate logic (the "WHERE clause" in the SQL SELECT statement).

For these reasons, one-for-one conversion is not usually recommended. Using a relational DBMS to simulate record-at-a-time file systems or DBMSs does not exploit its strengths. However, the one-for-one conversion approach may be the best choice for your program that runs every Sunday morning at 2 AM, especially if its performance isn't critical and it's the lowest cost approach per line of code.

Test Database Extraction

During application test, the individual program will probably be initially tested against minimal amounts of test data. For an integration test, it is necessary to operate against more comprehensive volumes of data. If a small but complete test database exists under DL/I, then the conversion procedures that will be used for the production database will also apply for converting the DL/I test data to SQL/DS test data. In fact, such a procedure will provide useful validation of the conversion process.

Where a DL/I test database does not exist and the production database is large, it will probably be expedient to extract a sample for development and testing. Such an extract must also be carefully thought out and the conversion program must be designed to accept it. Data extracted for test must be consistent. That is, foreign keys must in fact refer to existing primary keys after the extract.

Determine the sampling technique. This may be based upon such simple methods as every 100th database record. Make sure that there is a representative range of data complexity.

For large databases, the conversion process may need to be broken into steps which allow portions of the database to be unloaded independently. This would allow sections of the application to be tested without conversion of all segment types.

Unload Frequency

One technique for validating the complete conversion process is to run a set of transactions against both the DL/I data and the SQL/DS data. The DL/I data may then be extracted and loaded into SQL/DS format. It is then possible to unload both databases and run a compare program against the unload files. This implies that the DL/I unload process will be run numerous times. In other cases, data may be extracted only once to begin the development/testing phase, then not extracted again until the actual cutover. If the unload is to be run multiple times, there must be assurance that it can be run in a reasonable period. An unload requiring a three-day weekend that must be run after each test will significantly lengthen the time required for data conversion. Such problems will tend to favor a sampling extract program for test, and will justify additional time in the extraction development process.

For such cases, other techniques may also be applied. The DL/I data may be dumped with a high speed disk dump and restored to another DL/I database to allow for unload while the "real" DL/I data becomes available for activity. This will require additional DASD space, but may be very effective.

6.1.5 Creation of SQL/DS Objects

At some point prior to loading of the SQL/DS data, it is necessary to use SQL to create the SQL/DS data objects. Normal SQL/DS design parameters will apply to the development of the objects. The detail names and relationships will be present in the inventory of the data.

6.2 Loading Tables

Once the data has been extracted from the DL/I database, it may need further manipulation before it can be loaded into SQL/DS table(s).

Prior to loading, any data using a clustering index should be sorted into that sequence. Furthermore, provision should be made in the load time to allow for error correction. There will probably be exceptions discovered during the load which will have to be resolved. These can arise from a number of conditions and will require a variety of responses:

- The data was incorrect in DL/I
  - Correct the data in the DL/I database to permit test comparisons of the two databases and to ensure that any subsequent reports against the data are correct.
  - Correct the data in the SQL/DS table only to avoid changes in user outputs from the existing DL/I data.
  - Ignore the error wherever it will not prevent the loading of the table on the premise that data has the same validity that it had before in the DL/I system.
- The mapping does not always work (column overflows, etc.).
  - Change the data to permit mapping (assumes the DL/I data was not completely consistent with the application definition).
  - Change the mapping with corresponding redefinition of tables (i.e., DROP/CREATE definitions).
- Plan for inconsistencies
  - Force defaults that are acceptable.
  - If multiple inputs exist, select "most probably correct".
  - Submit to owner for resolution.

Many conditions will not be amenable to automated resolutions and even those that are may require review. Depending on the conditions uncovered and the technique for correction, it may be necessary to redo the conversion process starting with the unload of a corrected database or with corrected programs.

Some cases requiring intermediate processing are:

- When date fields in DL/I become DATE type columns in SQL/DS.

  The SQL/DS DBSU DATALOAD function supports three date formats. If the extracted field does not match one of those formats, it must be manipulated to produce a field that does. Since standard DATE columns carry the four-digit year (i.e., "1989" instead of "89"), it is likely that this will need to be added to the extracted data. For those applications where the date may span centuries and the application logic determines the correct century from two digit years, this logic (e.g., if year is greater than current year, then century is 18) must be incorporated into the conversion.
- When redundant data is to be eliminated, it may be desirable to identify and resolve all of the cases where the copies do not agree and take some action. In order to do this, all copies of the data will have to be extracted